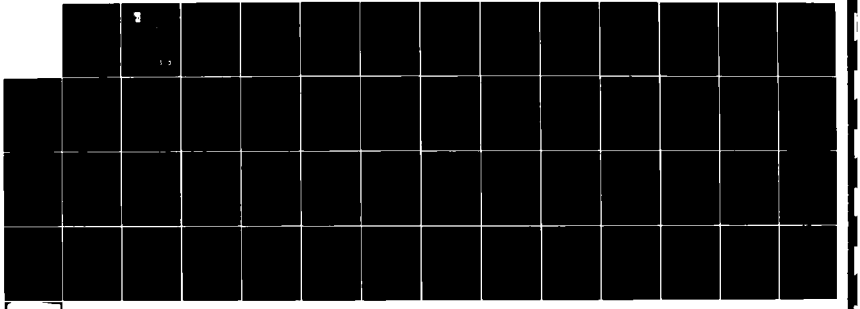
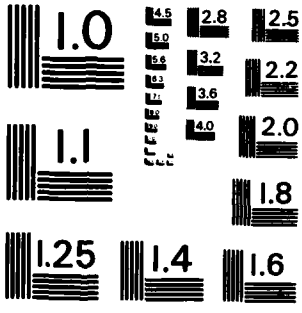
AD A124 258 SOFTWARE COST ESTIMATING(U) ROYAL SIGNALS AND RADAR ESTABLISHMENT MALVERN (ENGLAND) M STANLEY 13 MAY 82 RSRE-MEMO-3472 DRIC-BR-84024 1/1

UNCLASSIFIED F/G 9/2 NL



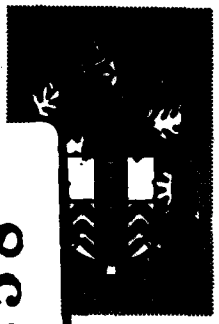
END
DATE FILMED
3 83
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS - 1963 - A

UNLIMITED

BR84024



AD A124258

RSRE
MEMORANDUM No. 3472

ROYAL SIGNALS & RADAR ESTABLISHMENT

SOFTWARE COST ESTIMATING

Author: M Stanley

PROCUREMENT EXECUTIVE,
MINISTRY OF DEFENCE,
RSRE MALVERN,
WORCS.

UNLIMITED

ROYAL SIGNALS AND RADAR ESTABLISHMENT

Memorandum 3472

Title: SOFTWARE COST ESTIMATING

Author: M. Stanley

Date: 13 May 1982

SUMMARY

The objective of software cost estimating is to determine what resources will be needed to produce and to maintain the software associated with a project. Resources of particular interest in software cost estimating are manpower, computer time and elapsed time. A good estimate will also show when and how costs will be incurred, so that the estimate can be used, not only to provide justification for software development, but also as a management control tool.

This report discusses why software cost estimating is error prone, and summarizes a number of different strategies for software cost estimating inherent in published models of the software development process, commenting on their strengths and weaknesses. Nineteen published models are considered.

SOFTWARE COST ESTIMATING

SUMMARY

The objective of software cost estimating is to determine what resources will be needed to produce and to maintain the software associated with a project. Resources of particular interest in software cost estimating are manpower, computer time and elapsed time. A good estimate will also show when and how costs will be incurred, so that the estimate can be used, not only to provide justification for software development, but also as a management control tool.

This report discusses why software cost estimating is error prone, and summarises a number of different strategies for software cost estimating inherent in published models of the software development process, commenting on their strengths and weaknesses. Nineteen published models are considered.

Author: M.Stanley
Date: April 1982

CONTENTS

1.0 INTRODUCTION. 1
2.0 PROBLEMS IN SOFTWARE COST ESTIMATING. 1
3.0 ESTIMATING METHODS. 2
4.0 TYPES OF PARAMETRIC MODEL. 3
4.1 Regression Models. 3
4.2 Heuristic Models. 3
4.3 Phenomenological Models. 4
5.0 GENERAL PATTERN FOLLOWED BY THE MODELS 4
5.1 Estimate Software Size. 4
5.2 Convert Size Estimate To Labour Estimate. 5
5.3 Adjust Estimate For Special Project Characteristics. 5
5.4 Divide The Total Estimate Into The Different Project Phases. 6
5.5 Estimate Non Technical Labour Costs And Costs Of Computer Time. 6
5.6 Sum The Costs. 6
6.0 OBJECTIVES OF COST ESTIMATING MODELS. 6
7.0 HISTORIC DATABASE OF SOFTWARE COSTS. 8
8.0 FACTORS INFLUENCING SOFTWARE COSTS. 9
8.1 Project Specific Factors. 9
8.1.1 Size Of The Software. 9
8.1.2 Percentage Of The Design And/or Code Which Is New. 10
8.1.3 Complexity Of The Software System. 10
8.1.4 Difficulty Of Design And Coding. 11
8.1.5 Quality. 11
8.1.6 Languages To Be Used. 12
8.1.7 Security Classification Level Of The Project. 12
8.1.8 Target Machine. 12
8.1.9 Utilisation Of The Target Hardware. 12
8.1.10 Volatility Of The Requirement. 12
8.2 Organisation Dependent Factors. 13
8.2.1 Project Schedule. 13
8.2.2 Personnel. 14
8.2.2.1 Technical Competence. 14
8.2.2.2 Non Technical Manpower. 14
8.2.3 Development Environment. 15
8.2.4 Resources Not Directly Attributable To Technical Aspects Of The 16
8.2.5 Computing Resources. 16
8.2.6 Labour Rates. 16
8.2.7 Inflation. 17
9.0 ITEMS COVERED BY THE MODELS. 17
9.1 Project Phases Covered. 17
9.2 Non Technical Manpower Costs. 17
9.3 Costs Other Than Manpower. 18
9.4 Cost Of Associated Software Development. 18
10.0 PUBLISHED MODELS. 18
10.1 Number Of Parameters. 19
10.2 Confidence In The Models. 19
10.3 Actual Models. 19
11.0 CONCLUSIONS. 20

CONTENTS

APPENDIX A COSTING MODELS.

A.1 THE AEROSPACE MODEL. A-1
A.2 THE GRC MODEL 1974. . . . . A-1
A.3 THE GRC MODEL 1977. . . . . A-2
A.4 THE ESD MODEL 1975. . . . . A-2
A.5 THE ESD MODEL 1978. . . . . A-3
A.6 THE SDC MODEL. A-4
A.7 THE WOLVERTON AND MODIFIED WOLVERTON MODELS. . . . A-4
A.8 THE IBM MODEL. A-5
A.9 THE TECOLOTE MODEL. A-5
A.10 THE NADC MODEL. A-6
A.11 THE TRW "SCEP" MODEL. A-6
A.12 THE HALSTEAD MODEL. A-7
A.13 THE BOEING MODEL. A-7
A.14 THE GRUMMAN MODEL, SOFCOST. A-7
A.15 THE DOTY MODEL. A-7
A.16 RCA PRICE-S MODEL. A-8
A.17 THE PUTNAM MODEL. A-9
A.17.1 The Phenomenological Aspects Of The Model. . . . A-9
A.17.2 Empirical Assumptions. A-10
A.17.3 The Model Processing. A-11
A.18 THE JPL MODEL. A-12
A.19 THE SLICE MODEL. A-12

APPENDIX B COMPARISON OF COST MODELS

APPENDIX C HISTORIC DATABASE - DATA REQUIRED.

C.1 INTRODUCTION. C-1
C.2 DATA REQUIRED. C-1
C.2.1 Project Size. C-1
C.2.2 Volatility Of The Requirement. C-2
C.2.3 Application Mix. C-2
C.2.4 Language. C-2
C.2.5 Percentages Of New Design And Of New Code. . . . C-3
C.2.6 Hardware/software Design Interactions. C-3
C.2.7 Hardware Utilisation. C-3
C.2.8 Platform Type. C-3
C.2.9 Experience Of The Development Organisation . . . C-4
C.2.10 Characteristics Of The Development Organisation: C-4
C.2.11 Software Tools And Techniques Employed: C-4
C.2.12 The Project Development Schedule. C-5
C.2.13 Costing Information. C-5
C.2.14 Resource Constraints. C-5
C.2.15 Project Cost. C-6
C.2.16 Start Year. C-6
C.2.17 Other Data. C-6

APPENDIX D HISTORIC DATABASES IN THE USA

APPENDIX E REFERENCES

1.0 INTRODUCTION.

The objective of software cost estimating is to determine what resources will be needed to produce and to maintain the software associated with a project. Resources of particular interest in software cost estimating are manpower, computer time and elapsed time. A good estimate will also show when and how costs will be incurred, so that the estimate can be used, not only to provide justification for software development, but also as a management control tool.

The purposes of this report are to explain why cost estimating is error prone, to summarise a number of different strategies for software cost estimating inherent in published models of the software development process, and to comment on their strengths and weaknesses. Nineteen published models are considered. They are described in Appendix A.

2.0 PROBLEMS IN SOFTWARE COST ESTIMATING.

Many military software projects are essentially innovative, and "both history and logic suggest that cost overruns are endemic to innovative engineering" (Ref. 1). Software cost estimating is a complicated process because project development is influenced by a large number of variables many of which are subjective, non-quantifiable and interrelated in complex ways.

Some reasons for not obtaining a good estimate are:

1. a lack of understanding of the process of software development and maintenance;

2. a lack of understanding of the effects of various technical and management constraints;

3. a view that each project is unique, which inhibits project to project comparisons;

4. lack of historic data against which the model can be checked:

5. lack of historic data for calibration. The process by which a model is fitted to a given cost estimating situation is called calibration. The calibration of a model may be performed using formal curve fitting methods on a representative historical data set, by selecting values from experience, or, with some models, by running the model in calibration mode, to assign values to selected parameters using appropriate historic data.

In addition to these reasons, current estimating techniques suffer from:

1. inadequate definition of the objective of the estimate, (whether it is intended as a project management tool or needed purely to aid in making a decision as to whether to go ahead), and at what stage the estimate is required, so that inputs and outputs can be chosen appropriately;

2. inadequate specification of the scope of the estimate (what is included and what is excluded);

3. inadequate definition of and understanding of the premises on which it is based.

3.0 ESTIMATING METHODS.

There are broadly five methods of estimating the cost of developing and of maintaining a software system:

1. seat of the pants and personal experience, using analogy with similar historic projects and extrapolation, based mainly on size and difficulty;

2. the constraint method;

3. percentage of hardware method;

4. simulation;

5. parametric modelling.

The first method, (sometimes refered to as a WAG, or "wildly aspiring guess"), has been and still is very popular because no better method has been proven. One of its problems is that each estimate is based on different experience, and therefore different estimates of the cost of a single project may vary very widely. A second problem is that the estimator must have experience of a similar project, of a similar size. Experience does not work on systems larger than those in the base used for comparison, nor on systems with a totally different content.

The constraint method is equivalent to taking an educated guess. Based on schedule, cost or manpower constraints, a manager agrees to develop the software within the constraints. The constraints are not related to the complexity of the project. In general, this method will result in delivery of the software within the specified constraints, but with the specification adjusted to fit the constraints.

The percentage of hardware method is based on two assumptions:

1. software costs are a fixed percentage of hardware costs;

2. hardware cost estimates are usually reasonably accurate.

The Doty study (Refs 25 and 26) indicated that the first of these assumptions is not justified.

Simulation is widely used in estimating life cycle support costs for hardware systems, but it is not appropriate for software cost estimating because it is based on a statistical analysis of hardware failure rates and spares logistics for which there is no software equivalent.

The models considered by the author are all parametric models (sometimes called SWAG, or "Scientific WAG"). It has been suggested that there is some universal model or formula which represents all types of software and all organisations. However, each estimator must tailor the universal model to take account of the idiosyncracies of his own organisation and the characteristics of his own application. This can be a difficult task since organisations are not static but change with time.

4.0 TYPES OF PARAMETRIC MODEL.

Parametric models (SWAG) may be divided into three classes (Ref. 9). The models considered do not necessarily fit tidily into one or other of the three classes, but if a model fits more into one class than into either of the others it will be regarded as a member of that class. The classes are:

4.1 Regression Models.

The quantity to be estimated is mathematically related to a set of input parameters . The parameters of the hypothesised relationship are arrived at by statistical analysis and curve fitting on an appropriate historical database. There may be more than one relationship, to deal with different databases, different types of application and different developer characterestics.

4.2 Heuristic Models.

In a heuristic model observation and interpretation of historic data are combined with supposition and experience. Relationships between variables are stated without justification. The advantage of heuristic models is that they need not wait for formal relationships to be established describing how the cost driving variables are related. Over a period a given model can become very effective in a stable predicting environment. If the model fails it is is adjusted to deal

with the situation. It therefore becomes a repository for the collected experience and insights of the designers.

4.3 Phenomenological Models.

The phenomenological model is based on a hypothesis that the software development process can be explained in terms of some more widely applicable process or idea. For example, the Putnam model is based on the belief that the distribution of effort during the software life cycle has the same characteristics as the distribution of effort required to solve a given number of problems given a constant learning rate. This has been shown to follow the Rayleigh distribution.

5.0 GENERAL PATTERN FOLLOWED BY THE MODELS

Most of the estimating models considered by the author follow a similar pattern, based on the following six steps:

1. estimate software size;
2. convert size estimate to labour estimate and possibly also a money estimate;
3. adjust estimate for special project characteristics;
4. divide the total estimate into the different project phases;
5. estimate non technical labour costs and costs of computer time;
6. sum the costs.

Not all steps occur in all models (for example some models do not initially perform a total project labour or cost estimate, but start by estimating the different phases separately, so step 4 aggregates the separate estimates instead of dividing up the total estimate. Similarly the adjustments for special project characteristics may occur between steps 1 and 2 as well as or instead of between steps 2 and 3).

5.1 Estimate Software Size.

Most models start from an estimate of project size, although some models include algorithms for computing size from various other system characteristics, such as units of work.

5.2 Convert Size Estimate To Labour Estimate.

Some models convert from size to labour, other go directly from size to money estimates. In regression models, these conversions are either derived from productivity measures using the "cost per instruction" type of equation or they are derived using the "general summing equation". The "cost per instruction" equation has the form:

e=a*s**b+c

where * is multiply; ** is raise to the power:
e= effort needed, for example manmonths of effort or cost in money terms, to develop the program;
s=size of project, for example number of machine level instructions:
and values are chosen for a,b and c by curve fitting on as large a historic database as possible. Different values of a,b,and c are appropriate to different development organisations, different project types, different sets of units for measuring e and s, and different items included in the estimates.

The "general summing equation" has the form:

e=a1*f1+a2*f2+a3*f3+....+ai*fi+.....+am*fm

where the ai are input parameters derived from the description of the software characteristics (including size) and the characteristics of the development environment, and the the values of fi are chosen by curve fitting on a suitable historic database.

In heuristic models the relationship of size estimate to labour estimate is not necessarily based on a statistically derived expression. In a phenomenological model the relationship would derive from the underlying theory.

5.3 Adjust Estimate For Special Project Characteristics.

In some models an effective size is calculated from the basic size estimate obtained in step 1, in others an effective labour or cost estimate is calculated from the estimates obtained in step 2. The effective estimate is an adjustment of the basic estimate intended to take account of any special project characteristics which make it dissimilar to the pattern absorbed in the underlying historic database. Such variations, which include the effect of volatility of the requirement; different software tools; difficulty above the level of projects in the database; or a different method of dealing with support costs, are frequently based on intuitively derived relationships, unsupported by statistical verification.

The adjustment may precede amalgamation of the costs of the different phases, or a single adjustment may be applied to the total.

5.4 Divide The Total Estimate Into The Different Project Phases.

Each model which deals with a project's schedule makes assumptions about the allocation of effort in the different project phases. The simplest assumption defines a percentage of the effort for each phase, for example the much quoted 40% design, 20% code, and 40% test rule. It should be noted that this rule is not universally agreed. Some research at GRC (Refs 6 and 7) shows that other percentages may be more appropriate, and the percentage in each phase may depend on other software characteristics. Some models assume that manpower allocation with respect to time follows a rectangular distribution, others that it follows a beta distibution (PRICE-S), or a Rayleigh distribution(Putnam). In general the assumptions on manpower allocation with respect to time are based on historic data. The effect of deviating from the historic patterns has not been considered.

5.5 Estimate Non Technical Labour Costs And Costs Of Computer Time.

Where these costs are explicitly included, they are often calculated as a percentage of the technical labour costs. Sometimes such costs are included implicitly because they were included in the database from which the model was derived.

5.6 Sum The Costs.

The non technical labour costs and the cost of computer time where these are included in the estimates are added to the technical costs of the different phases of the software life cycle to obtain an aggregated cost estimate.

6.0 OBJECTIVES OF COST ESTIMATING MODELS.

Any assumptions implicit in a model should be clearly stated. Any cost estimating model should exhibit the following characteristics:

1. the model should have well defined scope:
   (It should be clear which activities associated with the software life cycle are taken into account in the model and which are excluded. It should also be clear which resources: manpower; computer time and elapsed time, are being estimated, and whether costs of support software are included.)

2. the model should be widely applicable:
   (It should be possible to tailor a model to fit individual organisations, and types of software development.)

3. the model should be easy to use;
(Input requirements should be kept to a minimum, and output should be provided in an immediately useful format.)

4. the model should be able to use actual project data as it becomes available;
(Initial project cost estimates are likely to be based on inadequate information. As a project proceeds more accurate data become available for cost estimating. It is essential that any estimating model be capable of using actual data gathered at any stage in the project life to update the model and provide refined estimates, probably with a lower likely range of values than achieved initially.

Estimating is based on a probabilistic model. This means that an estimate is a number in the likely range of the quantity being estimated, and our confidence in the estimate depends on the likely range of the quantity being estimated. The better the information we have on which to base an estimate, the smaller the likely range and the greater the confidence.)

5. the model should allow for the use of historic data in calibration for a particular organisation and type of software;

6. the model should have been checked against a reasonable number of historic projects;

7. the model should only require inputs based on properties of the project which are well defined and can be established with a reasonable degree of certainty at the time the estimate is required;

8. the model should favour inputs based on objective rather than subjective criteria;

9. the model should not be over sensitive to subjective input criteria;

10. the model should be sensitive to all the parameters of a project which have been established as having a marked effect on the cost, and should not require input of parameters which do not have a marked effect on cost;

11. the model should included estimates of how and when the resource will be needed;
(This is particularly important if the estimates are to be used for resource allocation but also important if the results are given in financial terms since inflation needs to be taken into account.)

12. the model should produce a range of likely values for the quantity being estimated;
(It is important to realise that an estimate cannot provide a

precise prediction of the future. It must, of course, predict sufficiently closely to be useful, and to do this it should ideally be able to place bounds on either side of the estimate with a stated probability that the actual figures will lie within the stated bounds.)

13. the model should include possibilities for sensitivity analysis, so that the response of the estimates to variation of selected input parameters can be seen;

14. the model should include some estimate of the risk of failure to complete within the estimated time or cost.

7.0 HISTORIC DATABASE OF SOFTWARE COSTS.

No matter what method is used for estimating the cost of a software development, the estimate will be based on historic data, either collected and analysed systematically or haphazardly. If the data was collected haphazardly it will not be clear what assumptions are implicit in the database.

It is essential that a database of historic data on software project costs exists, so that model parameters may be calibrated and models checked. As long as each software project is considered as unique, a base of relevant past experience cannot be established. In order to build up a relevant database all software projects must be regarded as going through a basically similar process, with a standard set of tasks to be performed. Such tasks will include system requirements specification, system design, coding, testing, quality control, documentation, system integration, maintenance and training plus project management, project support and planning and control.

In addition to a standard set of tasks to be performed it is important that standard terminology be used to describe the tasks and the characteristics of the software system and of the development organisation.

Several databases have been built up in America. One, based on US projects, (mainly ground based military projects and management information systems,) is held by Rome Air Development Center at Griffiss AFB, and it has been used to check some of the current American cost estimating models. It does not include data on maintenance costs, nor on the cost of support software for development systems. Other databases available in the USA are listed in Appendix D.

Data currently available from UK MoD project costing does not normally separate software development costs from the cost of the hardware. Neither sufficient information on the nature of the software, any special factors to be taken into account, nor the resource allocation over the life of the project are included.

Appendix C shows a suggested list of data to be collected from each software project.

When creating a database of historic projects, it is important to be aware of any factors which may tend to distort the data. One such factor is the "Parkinson" effect, whereby a software development tends to fill the time, space and cost allocated to it. Thus the same requirement could result in totally different costs and size of software if one implementation were unconstrained and another was subject to strict limits.

8.0 FACTORS INFLUENCING SOFTWARE COSTS.

The factors which influence the cost of a software project may be divided into those which are contributed by the development and maintenance organisation, many of which are subjective and those which are inherent in the software project itself. Current models differ in respect of the factors that are required as specific inputs. Many different factors may be subsumed in a single parameter in some models, particularly the more subjective parameters.

The Doty study (Refs 25 and 26) collected data from a wide variety of different databases and analysed the effect on programmer productivity of many factors, (where productivity is defined as lines of delivered source code per manmonth). The effects were different in different application areas, and for different sizes of project. Some of the figures are quoted in this section, but for full details of this work the reader is referred to Refs 25 and 26.

Note: some factors, such as difficulty and complexity, though widely used, are ill defined. In some models they have precise meanings, but the meanings differ from model to model. In this section the terms are used in an intuitive sense rather than with precise meanings, unless a precise definition is given.

8.1 Project Specific Factors.

8.1.1 Size Of The Software.

A favourite measure for software system size is lines of operational code, or deliverable code (operational code plus supporting code, for example for hardware diagnostics) measured either in object code statements or in source code statements. It is rarely specified whether source code statements include non-executable code, such as comments and data declarations. Other measures are lines of code including non deliverable code (i.e. all code written during project development, whether deliverable or not, including support software and test software), number of functions to be performed or number of modules (where function and module are intuitive measures) or number of inputs and outputs.

Parametric cost estimating models relate cost in some way to the size estimate, and are therefore heavily dependent on the accuracy of the size estimate. In some models a reasonableness check of the size estimate is included. The size is indicated in more than one way and the different estimates are checked for consistency. One way to do this is by indicating the class of function to be performed by the software system and the languages to be employed. A database of past projects is then searched, to see what range of sizes have previously been achieved for that type of software. Most databases so far available do not include the size or cost of non deliverable software.

Another size indicator which is sometimes used is related to the size and structure of the database to be handled by the operational software.

Some models include algorithms for computing project size, either by splitting the project into smaller parts, or by analogy with other similar software. Other models use size estimates as the starting point.

The Doty study showed that size estimated can vary by as much as 10 times if the units are not well defined. Since some support software, test software and diagnostic software may be non deliverable although developed on the project, size estimates for the same project will be very different if such software is included in one size estimate and not in another. Similarly, if a model makes an assumption about the proportion of the total software which is included because it assumes a certain percentage of software for simulation of the environment, or other non operational software, this should be clearly stated. The Doty study found that deliverable code averaged 70% of total code developed, but with a standard deviation of about 30%.

8.1.2 Percentage Of The Design And/or Code Which Is New.

This is relevant when moving existing software systems to new computer hardware, when planning an extension to or modification of an existing software system, or when using software prototypes.

8.1.3 Complexity Of The Software System.

It is recognised in the software industry that different software projects have different degrees of complexity, usually measured by the amount of interaction between the different parts of the software system, and between the software and the external world. The complexity affects programmer productivity, and is an input parameter for several models.

8.1.4 Difficulty Of Design And Coding.

Different application areas are considered to have different levels of difficulty in design and coding, affecting programmer productivity. For example operating system software is usually regarded as more difficult than stand-alone commercial applications.

Software projects might be given a difficulty or an application mix rating, according to the degree to which they fall into one (or more) of the following application areas:

1. real time systems;
2. operating systems;
3. self contained real time projects;
4. stand alone non-real time applications;
5. modifications to an existing software system;
6. rewrite of an existing system for new target machine.

There are, of course, other categories. Each different model deals with the difficulty in its own way, some requiring estimates of the percentage of the software system which is of each type, others asking for a number on a predefined scale. Others merge the factor with the complexity rating.

8.1.5 Quality.

Quality, documentation, maintainability and reliability standards required are all included in a single factor. This factor is sometimes called the platform type, reflecting the fact that the documentation and reliability requirements for software in a manned space craft are higher than in a stand-alone statistical package. The documentation and reliability requirements may be given a defined numeric scale from 1 to 10 (say). In some estimating models there is also a parameter for the number of different locations at which the software will be run.

The Doty study found that as the number of pages of external documentation required per thousand lines of source code increased by 10%, programmer productivity decreased by 63%. If independent validation and verification were required then the cost of the software increased by about 20%.

8.1.6 Languages To Be Used.

The class of programming language used affects the cost, size, timescale and documentation effort. Some models require to know the percentage of the software in each language, others merely require input of the language to be used in the majority of the software. Some models only apply to projects using assembler code.

The Doty study estimates that the ratio of the total life cycle cost of a project programmed entirely in assembler to the total life cycle cost of the same project programmed entirely in a high level language could be as high as 5:1.

8.1.7 Security Classification Level Of The Project.

The higher the security classification of the project the more it will cost, because of the additional precautions required. The security classification is not an input factor in most models.

8.1.8 Target Machine.

A few models require information about the target machine, in particular the word length, and whether the machine has an established architecture.

8.1.9 Utilisation Of The Target Hardware.

Several models include a parameter for target hardware utilisation. As the percentage utilisation, either in terms of processor time or in terms of store space on the target machine. increase above about 50%, the estimated software development cost increases. The Doty study shows programmer productivity decreasing exponentially as the percentage of target hardware utilisation increases.

8.1.10 Volatility Of The Requirement.

The firmness of the requirement specification and the interface between developer and customer affects the amount of rework which will be needed before the software is delivered. This is a highly subjective factor, but nonetheless an important one, which is an input factor to several models.

The following are included in this factor:

1. amount of change expected in the requirement;

2. amount of detail omitted from the requirement specification;

3. concurrent development of associated hardware, causing changes in the software specification;

4. unspecified target hardware.

(The Doty study indicated that productivity can decrease by up to 50% if requirements are vague rather than detailed).

8.2 Organisation Dependent Factors.

The type of organisation responsible for software development has an influence of the software costs. Among the factors contributed by the organisation are:

8.2.1 Project Schedule.

Software projects cannot be speeded up to occupy less than some minimum time. Attempts to compress timescales below the minimum by applying more people to the project prove counter productive, since more time and effort are expended in communication between members of the project team than can be gained by adding extra people. There must therefore be either a minimum time below which the project cannot be completed, or at least a time below which the costs of saving a small amount of time become prohibitive. Conversely, if more time is allocated to a project than is required, it has been argued (by Putnam see refs 30 to 37) that the cost decreases. However, other models, (e.g. the PRICE model, see refs. 16 to 22) show costs increasing if more time than some optimum time is allocated, because more manpower is consumed.

One effect of the compression of timescales is that work which should be done in series is undertaken in parallel, with the increased risk that some of the work will have to be scrapped (e.g. if coding is started before design is complete).

Not all models deal with project schedules. Of those which do, some assume the 40-20-40 rule (40% design, 20% coding, and 40% testing), and others use more elaborate scheduling assumptions. Some research by GRC (Refs. 6 and 7) throws doubt on the validity of the 40-20-40 rule, and indicates that phases are strongly inter-related, so that effort skimped in one phase will probably result in a considerable increase in the effort needed in a later phase. GRC emphasise that the database they use is small, and the results are therefore only preliminary.

Few models require input of a proposed development schedule. In the PRICE-S model the proposed development schedule is an optional input. Some models actually deliver a recommended, or assumed development schedule.

8.2.2 Personnel.

The personnel assigned to a project contribute to the cost depending on how maManpower levels.

Most projects are resource limited, in that the number of people with a given skill who are available to the project is limited. The level of manpower available at any stage in a project will affect the timescales, and hence the cost, but it is not a required input for most models.

Research by ESD indicates that projects show a resource consumption having a rectangular distribution with respect to time. This is at variance with IBM data published by Aron, and with Putnam's model which shows resource consumption having a Rayleigh distribution. The PRICE model uses a Beta distribution for each phase in the development. Aron has suggested that the use of modern techniques will cause the distribution to flatten out. There is considerable data in the US regarding the relationship of development time and resource distributions to program size, but almost no data on the effect of deviating from the historic pattern.

8.2.2.1 Technical Competence.

Several statistical analyses (Refs 8 and 25) have shown that for large projects, the experience of the technical personnel makes very little difference to their productivity. However several models use the technical expertise of the personnel and the experience of the developer on similar applications as an input, although this is a highly subjective judgement.

The Doty study indicates that an extra 6 months should be added to the development time for personnel to gain experience of new hardware, or an unfamiliar application area.

8.2.2.2 Non Technical Manpower.

Estimates of the non technical manpower levels required by a project are frequently made as a percentage of the technical manpower levels, although some models compute non technical manpower requirements as a function of project size, some use the documentation and maintenance requirements to derive a relationship between technical manpower requirements and the manpower requirements for support

personnel and some ignore the question.

8.2.3 Development Environment.

The adequacy of the development environment, both in hardware and software depends largely on the management of the development organisation. This factor is not usually requested as an explicit input to a model, but may be implicit in the calibration of the model, or in some general management parameter. Three aspects of the development environment which are sometimes required as inputs to models are:

1. development machine:

   (The adequacy of the development machine as a host for developing software for the selected target, and the availability of the development machine to the software development personnel will affect both the schedule and the cost of a software development. The Doty study showed that time-sharing, where the development machine is constantly available, is 20% more productive than batch systems for software development.)

2. availabilty of associated software and hardware:

   (Projected late delivery of some item of associated hardware or software can affect schedules and costs. Models which use development schedules cater for this by rerunning with different input parameters.)

3. software tools and techniques to be used during system design and development.

   (That newer tools and techiques properly applied can reduce development effort has been demonstrated. Aron of IBM estimates that savings of up to 40% can be achieved by the application of modern techniques. Among techniques to be considered are:

   1. host/target development systems;
      (The Doty study indicates that development on a different machine from the target has an adverse effect on cost unless the target has inadequate facilities for software development.)

   2. high level languages;

   3. MASCOT or a similar design and test methodology;

   4. prototyping;

5. development libraries;

6. development database;

7. word processors for documentation;

8. adequate programming support environment.

The list is not complete.)

8.2.4 Resources Not Directly Attributable To Technical Aspects Of The Project.

The management style of the development organisation affects the amount of effort expended in communication between team members (meetings) and the level of non technical effort involved as well as the cost of bought-in software/hardware tools, costs of sub-contracting and profit. These factors are usually either ignored, or are implicit in the database from which the model is derived, or the are taken care of by a general management factor.

The geographical distribution of the development organisation may affect costs because of travel costs and the cost of transmitting data between sites, and is therefore input to some models.

8.2.5 Computing Resources.

Many models ignore the cost of computing, (hardware and computer time). Others, such as Wolverton, assume a fixed percentage of the total cost will be consumed by this item, while yet others take an average cost per manmonth with different average costs for each phase in the development.

8.2.6 Labour Rates.

If the model estimates costs in terms of money, rather than manhours, the relationship of labour costs to manhours within the development organisation may be required by the model. The model may be capable of reflecting increased rates for staff required to work unsocial hours because of decreases in the development timescale, or lack of availability of development tools.

8.2.7 Inflation.

Costs estimated in terms of money rather than manhours may take inflation rates into account, as well as the costs in a base year. If inflation is not built into the model, the rate of inflation may be required as an input.

9.0 ITEMS COVERED BY THE MODELS.

9.1 Project Phases Covered.

The majority of existing models cover only the cost of software development, from agreement of a requirement specification to delivery of the software to the customer. The lack of adequate information about a project prior to the existence of a requirements specification will usually preclude making more than a very crude estimate of size and cost at that stage.

Lack of an adequate database of historic software maintenance costs has hindered development of models for the complete life cycle. Several different sources give estimates of maintenance as a percentage of the total life cycle cost, varying from 40% (Putnam) to 80% (ref 41). Some other models (ref 15) estimate the annual maintenance effort from an equation of the form:

$$e=k*s^{m}+n$$

where * means multiply,
e=annual maintenance effort: s=project size:
k is a constant;
m depends on the number of years after delivery
and n is a constant.

Other models use a simple rule giving maintenance effort per thousand object instructions per year of maintenance. PPICF-SL uses the output from the PRICE-S model (which computes development cost and schedule) to compute life cycle costs. The algorithms and assumptions are not published. The Putnam model, which uses a fixed 40% of the total life cycle cost for maintenance, computes the costs on a month by month basis, based on the Rayleigh distribution of manpower over the whole life cycle.

9.2 Non Technical Manpower Costs.

The development cost estimates usually include the cost of documentation, frequently as a fixed percentage of the estimated technical effort needed for coding. The cost of design and of testing is also sometimes computed as a fixed percentage of the estimated manpower for coding. Some models included management and project support effort. Others do not.

9.3 Costs Other Than Manpower.

Where costs are given in terms of money the models sometimes include the costs of computer time, the costs of support hardware and bought in software, bought in labour and overheads. Other models only include direct labour costs.

Inflation is a factor in some models which include a development schedule.

9.4 Cost Of Associated Software Development.

When software size is estimated it may include:

1. operational software delivered to the customer;
2. support software needed for maintenance and development, delivered to the customer;
3. support software needed for maintenance and development, not delivered to the customer;
4. code created by the developer but not delivered to the customer, such as special test software, simulations and test tools;
5. an allowance for code which will be thrown away because of changes to or misunderstandings of the requirement:
6. support software such as operating systems, compilers, word processing software etc.

Models do not always indicate clearly which of the above are included in the cost estimates. Support and non deliverable software is sometimes covered in the cost estimates even when excluded from the size estimates. However, the ratio of delivered code to total code developed, or of support software to operational software has a wide variance according to the type of customer and the type of application. In particular, projects vary widely in the quantity of software needed for simulation of the operational environment.

10.0 PUBLISHED MODELS.

10.1 Number Of Parameters.

The models vary in the number of parameters which are required. For example, the IBM model used 29 paramters found in the IBM study by Walston and Felix (Ref 4) to be highly correlated with programmer productivity. A number of other models (e.g. GRC, Doty, TRW, Tecolote and Aerospace) provide productivity data using far fewer parameters.

The Putnam type of model uses only a few specific parameters together with a mathematical model calibrated by using industry data to provide estimates of the effects of changing planned duration or effort.

10.2 Confidence In The Models.

The statistical confidence with which the models may be used is quite low. They are presented, not as tested, well proven tools, but as guidelines and for information to back up additional techniques.

Probably the most thoroughly tested models are the PRICE model and the Putnam model. These both claim to have been verified with a fairly extensive American database, but the European experience seems to be very limited.

10.3 Actual Models.

The following models are presented very briefly in Appendix A with a comparison of the models in Appendix B:

1. the Aerospece model;
2. two GRC models;
3. two ESD modes;
4. the SDC model;
5. the Wolverton and revised Wolverton models;
6. the IBM model;
7. the Tecolote model;
8. the NADC model;
9. the TRW "SCEP" model;
10. the Halstead model;

11. the Boeing model;

12. the Grumman model (SOFCOST);

13. the Doty model;

14. the PRICE models;

15. the Putnam model (SLIM);

16. the JPL model;

17. SLICE.

11.0 CONCLUSIONS.

The work performed so far on software cost estimating has resulted in several different models, most of which have very limited applicability. Two models are known to be commercially available, PRICE-S and SLIM, and both claim to have been verified by use on a wide variety of US projects. They need more detailed study, and will be the subject of a separate paper.

The JPL model, although not commercially available, probably merits further consideration, and the ideas in SLICE for adjusting a model to differing views of the software life cycle are of interest.

The factors influencing software costs have been fairly extensively studied in the USA, and some measure of agreement has been reached on which factors are strongly correlated with cost, although there is less agreement how the effects should be measured or used in the models. The Doty study analyses these factors.

There is a need to accumulate relevant data in the UK against which any software costing models can be checked, and from which models can be calibrated. The terms used and the scope of any data collected must be rigorously defined, to prevent misuse and misinterpretation. Appendix C indicates the type of data which is needed, but the collection of the data needs to be done using a standard questionnaire, so that the collection is uniform. It will be difficult to collect such data on historic projects, but urgent action is needed to start the collection of relevant data on current and future projects.

When models are being assessed or developed it is important to check that the scope of the estimates are well defined, and that the input parameters, (particularly the size measure) are unambiguously defined. All models are very sensitive to size estimates, so definition of the units in which size is measured, and scope of software to be included in the size estimate are crucial.

CONCLUSIONS.

It is important to remember that, however well founded the model, the estimates output from it are based on estimates of size of software, which are also difficult to make, and subject to large errors. A model which permits the input of size as a range a possible values is therefore easier to use than one which demands a single figure.

If the purpose of the estimates is to assist management in comparing different proposals, some form of risk estimate, as in the Putnam and JPL models is very useful. A sensitivity analysis showing the effect on the estimates of varying a particular factor is not so essential, since a rerun of the model with different parameters can achieve the same result, but it is none the less a useful feature. Time based resource allocation is also a useful output from a model.

Very few of the models include any estimates for the total life cycle, because the data on life cycle costs is so sparse. Those models which do make estimates of the cost of maintenance are therefore based on relatively crude assumptions.

APPENDIX A

COSTING MODELS.

Descriptions of the models sometimes refer to the general model, which is described in section 5.

A.1 THE AEROSPACE MODEL.

Developed in 1975 by Aerospace Corporation this is a special case of the cost per instruction model, with c=0. Two values of a and b were calculated with e=manmonths of effort to develop the program and s=number of machine level instructions

The Aerospace model gives one pair of values to a and b for real time systems, and another for all other software systems.

A.2 THE GRC MODEL 1974.

The first GRC model, developed in 1974 by General Research Corporation, computes basic cost from the general equation with c=0. The model then modifies the basic cost to take account of target computer utilization. If u% of the processor capacity (store or speed) will be needed, and d is the adjusted cost estimate, then

d=e*0.7/(square root of(u-0.5)) when u>0.5

GRC also indicated two factors which were not taken into account, but which they felt to be significant. They were:

1. degree of difficulty or complexity
2. language (assembler code is assumed).

GRC state that the choice of a high level language will reduce basic cost by up to one third, but will adversely affect utilization by up to three times.

A.3 THE GRC MODEL 1977.

The second GRC model was developed in 1977 by General Research Corporation, as a result of work with the 1974 model. This model consists of four stages:

1. size estimation, based on the number of functions, language used and application;

2. size adjustment, based on volatility of the requirement and target computer type;

3. coding productivity model, which computes the labour needed for the coding phase, taking into account program size, application mix(difficulty), hardware utilisation, language, and software tools and techniques;

4. model of the design, test and support costs, taking each of these as a percentage of the coding labour (analysis and design=36%of the total labour, coding=21% of the total labour and testing=43% of the total labour). Computer costs are assumed proportional to the technical labour costs in each phase. Management and documentation are assumed to add 28% to the total technical labour cost.

The model includes a maintenance cost estimate based on the expected (decreasing) error rate per line of code, with cost varying according to application type. The model also takes into account the relationships between the different phases of the software life cycle.

A.4 THE ESD MODEL 1975.

The summary of notes of a US Government/Industry Software Workshop, sponsored in 1975 by the Electronic Systems Division, Hanscom Air Force Base, included a list, agreed between the participants, (Ref. 17) of factors influencing software costs, and the guesses of the effects of those drivers on software development and maintenance costs.

It was felt at the Workshop that the number of delivered executable source code instructions was a better estimation parameter than the number of object code instructions. From this list the ESD model was developed.

First a basic cost estimate is arrived at, based linearly on the number of source code instructions, language level and whether the code will be time critical. The basic cost estimate is then subjected to various multipliers according to project type (operating system or straight forward application) and experience with the application area. It is assumed that documentation will take 10% of the total cost, manpower for testing will take 40% of the total cost and other testing costs will take a further 15%.

The hardware utilization is deemed to have an asymptotic effect on basic cost, with the cost tending to infinity as the utilization rises to 100%. The realism of the development schedule is another cost driver where the cost increases as the development schedule is accelerated.

Factors listed by ESD as subjective are those whose affect on cost has not been quantified, but which were deemed nevertheless to be cost drivers. The effect of the subjective factors such as project complexity, personnel competence, stability of the requirements, adequacy and stability of the development environment, level of software technology used on the project, size and structure of the database being handled by the software and management style of the development organisation all need to be measured by reference to an adequate database. In the absence of such a database the ESD model uses arbitrary factors to adjust the basic cost estimate.

A.5 THE ESD MODEL 1978.

Developed in 1978 by J.Duquette and G.Bourdon, (ref. 11) the model consists of five stages:

1. sizing, transforms the functional requirements and details of the target hardware to size in terms both of object instructions and of source code instructions:

2. manpower, converts size to labour required during development:
   (The development effort is estimated using a set of equations of the form:

   $$E=a*S**b$$

   modified by a multiplier based on 14 different environmental factors.)

3. schedule, calculates the recommended development time, T, using an equation of the form:

   $$T=(c*S)/(d+f*S**p)$$

   where c,d,f and p are constants;

   (Intermediated milestones are at fixed percentages of the development time after start of development.)

4. manloading, spreads the manpower over the development, assuming a Rayleigh distribution;
   (The life cycle requirements are dealt with by assuming, as in the Putnam model, that 39% of the life cycle effort is needed for development, and that the manpower will continue as a Rayleigh distribution until 2.38 times the development time has elapsed. After this time labour requirements are assumed constant for the rest of the life cycle.)

5. cost, derives cost from the labour distribution by multiplying the manpower estimate for each month by the fully burdened labour rate, (average for all types of labour, and including overheads and other indirect costs). The cost takes account of inflation.

A.6 THE SDC MODEL.

Developed in 1967 by The Systems Development Corporation, the model is based on work sponsored by ESD. It produces an estimate of cost per object code instruction, based on a linear equation in the following 12 parameters:

1. complexity, measured by the amount of design effort associated with transferring data between modules;
2. percentage of clerical instructions;
3. percentage of information storage instructions;
4. frequency of operation;
5. volume of deliverable documentation;
6. business application or other application;
7. new machine or new to the developers;
8. whether graphic displays are used;
9. whether random access storage (disc) is used;
10. percentage of the programmers who are involved in the design;
11. continuity of personnel;
12. number of locations used for program development.

A.7 THE WOLVERTON AND MODIFIED WOLVERTON MODELS.

Developed by P.W.Wolverton of TRW in 1973, and modified by the USAF Avionics Laboratory, (refs. 13,14 and 17) this model is based on the TRW proprietary data base containing historic information on cost per instruction. Built into the model is a set of standard cost equations, with c=0 and b=1 and different values of a according to three different difficulty ratings (measures of the amount of interaction expected), and six different categories, ranging from time critical processes to algorithms for performing simple logic or

mathematics. s is measured in machine code instructions and e in money. The functions are also divided into old code and new code.

The modified Wolverton model is a short computer program which uses the Wolverton cost/instruction, and generates costs for old and for new code, in each of 9 difficulty grades. The output is divided into cost for analysis (20%), design (18.7%), coding (21.7%), testing (28.3%) and documentation (11.3%).

Cost per instruction is, of course, peculiar to one particular organisation, and would need updating to take account of inflation. No allowance is made for timescale of the development, or for the effect of size on cost/instruction, except indirectly via the difficulty grades.

A.8 THE IBM MODEL.

The IBM model is documented in the IBM proprietary report "Estimating Software Life Cycle Costs" by J.C.Malone, April 1975. Equations are developed, using data from IBM projects which employed structured programming and the "chief programmer" technique. The equations are proprietary to IBM, and the model applies to projects requiring more than 25 programmers, more than 30,000 lines of deliverable code, more than 6 months development time and more than one level of management. The model is based on productivity, which is related to 29 different variables in the report in Ref 4, by C.E.Walston and C.P.Felix.

The model was applied by the System Evaluation Group of the USAF Avionics Laboratory to a number of projects (Ref. 16) and gave consistently lower estimates than other models. The actual costs of software developments were not available on the majority of the projects. On the two projects where actual costs were available, the IBM model performed better than average.

A.9 THE TECOLOTE MODEL.

Developed by B.C.Frederic of Tecolote Research Inc. in 1974, (ref. 12) as a provisional model, this is a set of equations developed specifically for estimating development costs of naval tactical software for fire control systems. The basic equations relate cost separately to each of 5 different input parameters and the user selects whichever of the 5 parameters he has most confidence in. The parameters are number of air/sea targets to be tracked, or total fast storage to be used, or total number of instructions in the operational software, or total number of instructions in the operational software plus the test software, or total manmonths of labour needed to produce the software.

The model is based on a very small, very specialized database and is not therefore generally applicable.

A.10 THE NADC MODEL.

Developed as a result of a study performed in 1971 by the US Naval Air Development Center, called "A cost by function model for avionic computer systems", this model consists of several equations for predicting costs of research, development, test, evalation and production of future avionic computer systems. The model consists of 10 basic steps for converting the functional requirements of the system into software and hardware requirements and costs. The general equation model is used for computing manmonths of effort to develop the software. It is a linear equation with the following inputs:

1. number of machine language instructions in the delivered program;
2. number of man-miles travelled by contractors;
3. number of document types produced;
4. number of independent consoles in the delivered system;
5. percentage of new instructions;
6. average programmer experience, in years.

No account is taken of development schedule, target machine utilisation or of the level of software development technology or the variations in difficulty of producing different types of code.

A.11 THE TRW "SCEP" MODEL.

Developed by B.W.Boehm and P.W.Wolverton of TRW Defense and Space Systems Group in 1978, the model uses the standard work breakdown structure (Ref. 14) to define which costs are included. The model was initially developed to study the impact of new software technologies. Input data includes size of software systems measured in source code instructions, number of data items, program type, complexity, language, percentage new code and target hardware utilization.

A.12 THE HALSTEAD MODEL.

Developed by M.H.Halstead in his book (Ref 47) published in 1977, this model relies on more detailed knowledge of the software to be developed that can possibly be available in the early stages of a project. For example, inputs to the model include number of uses of different operators and of operands within the program. This model will not therefore be discussed further.

A.13 THE BOEING MODEL.

Developed in 1977 by Boeing Computer Services Inc. under contract to Rome Air Development Center, this model was initially intended for the study of new software technologies. The system is divided into five types of software and the number of delivered instructions is estimated for each component. The system development effort is obtained using a different productivity rate for each type of software. The development effort is divided into 6 phases, using fixed ratios. The phase estimates are adjusted for certain development and application characteristics.

A.14 THE GRUMMAN MODEL, SOFCOST.

Developed in 1979 by H.F.Dircks of Grumman Aerospace Corporation, (refs. 2 and 3) this model consists of three phases. The first two phases produce a matrix of sizes of the constituent software modules for a project, according to the standard work breakdown structure (WBS). The sizes are based on a database, plus information from the project team. The final phase adjusts the size estimates to allow for choice of language, calculates the estimate of effort required using different parametric equations for each phase (design, code and test). The model next adjusts the productivity according to the values given to 30 parameters describing experience, complexity, tools and techniques and hardware. Costs and schedule are then calculated, divided into six areas: technical; support; management: documentation; configuration control and resources. The schedule is broken down into software life cycle phases and includes start and end dates.

A.15 THE DOTY MODEL.

Developed in 1977 by Doty Associates Inc. as a result of a study commissioned by RADC, this model is based on extensive analysis of the factors affecting software development costs. The model is described in Refs 25 and 26. The model divides cost into primary costs and secondary costs, such as cost of computer time, which are taken to be a fixed proportion, h, of the total cost. The primary costs are estimating by multiplying the development effort by the burdened labour

rate, L, including direct labour costs, associated overheads, administration and fees. Thus

cost=E*L(1+h).

The development effort is estimated using a set of equations of the form:

E=a*S**b

modified by a multiplier based on 14 different environmental factors.
The modifier, and the values of a and b, vary according to the application area, whether the project is over or under a given size and the percentage utilisation of the target hardware.

The recommended development time, T, is calculated using an equation of the form:

T=(c*S)/(d+f*S**p)

where c,d,f and p are constants.

Intermediated milestones and the percentage of the spend which is expected by each milestone are indicated in the model as a reasonableness check.

A.16 RCA PRICE-S MODEL.

Developed in 1978 by RCA, this model (refs. 16 and 18 to 22) is a heuristic model based on the RCA PRICE model for estimating hardware system development costs. The model provides facilities for estimating software development costs and development schedules. A second model, PRICE-SL is available for estimating software maintenance costs.

The model, which is available as a computer program, is marketed by RCA Corporation. It is available within MOD on a PRIME computer owned by MoD (DSWS) in Bath.

The algorithms and implicit assumptions on which the model is based are not published. RCA claim that the model has been verified on a very large database of American projects (of the order of hundreds of projects).

The model covers the development of the software in three phases: design; implementation; and test and integration. It divides the labour involved into five types: management; systems engineering: programming; documentation; and quality assurance and configuration control and allows for the specification of resource constraints. The costs are calculated basically in terms of money, with inflation taken into account, although the money may be converted into manhours or manmonths, if requested.

Two of the model parameters (one, called resource, used to describe the management style of the development organisation, and the other, called application mix, used to describe the type of

application) can be assigned values by using the model in calibration mode with appropriate historic data.

The model outputs include a development schedule (suggested by the model if not provided as an input). There are also two optional sensitivity analyses, varying either the application mix parameter with the project size, or the resource parameter with the project complexity parameter. The output is a table giving the effect of the variations on project cost and on total development time; The model also gives an optional comparison of the costs and length of time needed if working to the input development schedule compared with the cost and length of time needed if working to the suggested development schedule, and an optional monthly progress summary, giving, on a monthly basis the percentage of each phase of the project which should be completed, and the costs incurred in that month.

A.17 THE PUTNAM MODEL.

Developed by L.Putnam in 1974, (refs. 30 to 37) the model is available as a computer program running on the Hewlitt Packard HP-85 computer. It is marketed in the UK by PACTEL under the name SLIM. Putnam claims that his model has been verified on the large American software database held by the Rome Air Development Center, and it was developed from US Army software development data.

A.17.1 The Phenomenological Aspects Of The Model.

The Putnam model is based on the equation:

$$s=c*(k**\{1/3\})*(T**\{4/3\})$$

which is derived from the Rayleigh equation:

$$YDOT=\{k/(T**2)\}*t*\{e**\{(-t**2)/(2*(T**2))\}$$

for distribution of manpower within the project.
t=time since project development started;
YDOT=manpower on the project at time t, including both technical and non technical labour;
T=scheduled software development time;
k=effort required to develop and to maintain the software, including technical effort, management, documentation, quality assurance etc.:
s=size of the software in lines of source code;
c=the "technology constant", which is used to describe the characteristics of the development organisation (the development machine throughput and the software engineering tools and techniques) and the application class of the software project. The value of the technology constant for a given development organisation developing a particular class of software can be calculated by running the model in calibration mode, with appropriate historic data.

Putnam's assumption of a Rayleigh distribution is based on the belief that the distribution of effort during the software life cycle exhibits the same characteristics as the distribution of effort

required to solve a given number of problems given a constant learning rate. The belief is backed up by analysis of historic data on US Army projects, mainly written in assembler code, and without the use of modern techniques. His findings are at variance with the findings of ESD from a different database, although Aron found the same sort of resource allocation in an IBM database. However Aron has stated that he expects the resource allocation profile to flatten into something like that found by ESD (constant throughout the development), as modern programming techniques are introduced.

A.17.1.1 Development effort and total life cycle effort.

The model allocates 40% of the life cycle effort (k) to development effort (E).

A.17.1.2 Scheduled development time (T).

In the Putnam model the productivity of the personnel is not considered to be a constant for a given project, in a given development environment. Productivity is assumed to vary as the fourth power of the scheduled development time. Thus the tighter the development schedule, the lower the productivity of the personnel, and so the higher the development cost. The scheduled development time is therefore a very important factor in determining the cost. The model initially calculates a minimum feasible scheduled development time (TMIN), and corresponding cost. Any schedule allowing more time for development than the minimum time is regarded as feasible, and will incur a corresponding lower cost estimate.

A.17.2 Empirical Assumptions.

The Putnam model contains a number of assumptions based not on the Rayleigh equation for development effort, but on some empirically verified assumptions.

A.17.2.1 Project difficulty.

Project difficulty, D, defined by the equation

$$D=k/(T^{**2})$$

has been found empirically to be a measure of the difficulty of the project which is a measure of the degree of parallelism required during project development, and of the volatility of the requirement.

The difficulty of the actual development may be decreased by relaxing the development schedule, without a corresponding decrease in the manpower allocation, but it cannot be increased above the inherent

project difficulty level, DMAX.

DMAX=kmax/(TMIN**2)

where TMIN is the minimum feasible scheduled development time, and kmax is the corresponding estimate of life cycle effort.

A.17.2.2 Difficulty gradient (G) of a project.

The model includes a difficulty gradient parameter which is a measure of the maximum rate of code production which can be achieved on the project because of interfacing problems, or degree of innovation in the project. The difficulty gradient parameter has been found empirically to be constant for a given project, and related to the scheduled project development time and to the total effort required over the project life cycle.

G>=2*k/T**3

A.17.3 The Model Processing.

The model calculates the overall project size, (mean and standard deviation), from input of size estimates (best estimate and range) of the constituent software modules. It takes the project size(s), and the difficulty gradient(G), and solves the simultaneous equations

s=c*(k**{1/3})*(T**{4/3})

G>=2*k/T**3

to calculate the minimum feasible scheduled development time, (TMIN), below which the project would not be feasible, and the corresponding cost (kmax). It varies the values of s and G about the mean values input, to obtain a distribution of fastest possible development schedules and corresponding costs.

Development effort, E=40% of life cycle effort(k). It is given both in manmonths and in money terms, optionally taking inflation into account. The model gives a range and standard deviation on the estimates, but effort is not broken down into technical and non technical (management, clerical etc.).

The user has the option of giving a proposed scheduled development time and cost, and asking the program whether it is feasible or not.

A.17.3.1 Manpower levels.

The model calculates the manpower on the project at time t for selected values of T and k. It computes the manpower levels needed at each stage of the life cycle using the Rayleigh equation given above. It assumes that manpower on the project follows a different Rayleigh curve at each of the stages: feasibility study; project definition; implementation; test and integration and maintenance. The curves overlap, and the model is only concerned with implementation (design

and coding, test and integration and maintenance. The consumption of effort is given on a month by month basis.

A.17.3.2 Variations on a theme.

Given c,s and G from the inputs to the model, the computer model allows the user several variations on the basic theme, such as specifying a value for k, and calculating corresponding T, or specifying a value for T and calculating corresponding k. It also performs a Monte Carlo simulation of each of the above stages, varying s and G and c if c is not input explicitly. This gives a distribution of values for T,k and the manpower per month, allowing for calculation of the probability of achieving a given value of T or k.

A.18 THE JPL MODEL.

Developed in 1981 by R.C.Tausworthe of the Jet Propulsion Laboratories, Deep Space Network Data Systems, the JPL model (refs. 28 and 29) modifies and combines a number of existing models, such as the GRC model, the Doty model and the Putnam model. The JPL model calibrates the task magnitude and difficulty, development environment and software technology effects through a prompted set of 50 questions. The model is based on the JPL database. It scales a standard work breakdown structure which is input to a PERT/CPM system to produce a detailed schedule and resource allocation for the project.

A.19 THE SLICE MODEL.

Developed in 1980 at IBM, SLICE (System Life Cycle Estimation) is a technique for software cost estimating described in reference 48. There is no program available for using the technique. The model takes project size, and an estimate of programmer productivity, and calculates the technical effort required to complete the project. The technical effort is converted to a time phased project plan using the crude assumption that if e**2 manmonths of effort are needed on a project, then the optimium staffing plan will be e men for e months.

The difference between SLICE and most cost estimating models is in the method of estimating programmer productivity. Most models assume a particular succession of phases through the life cycle. SLICE requires the user to describe his project life cycle in phases, and then assign a percentage of effort to each phase. The next input is a productivity figure (in lines of source code or in object code instructions per day). The user is assumed to know a productivity rating for his own organisation from examination of past projects, just as he is assumed to know the pattern of life cycle phases and the percentage of effort expended in each phase by his organisation from examination of past projects. The productivity figure may be an overall figure for the

whole project, or it may apply only to a subset of the phases. If it applies to a subset of the phases it is scaled to give a productivity figure for the total life cycle, on the basis of the percentage assigned to each phase. The technical effort required is then calculated by applying the adjusted productivity figure to the estimated project size.

APPENDIX B

COMPARISON OF COST MODELS

COMPARISON OF COST MODELS

******MODEL NUMBER**********	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
MODEL NAME	AEROSPACE	GRC1	ESDC1	SDC	WOLVERTON	IBM	TECOLDTE	NADC	TRW	HALSTEAD	BOEING	GRUMMAN	DOTY	GRC2	PRICE-M	PUTNAM	JPL	SLIC	ESD2
Date	:75	:74	:75	:67	:73	:75	:74	:71	:78	:75	:77	:79	:77	:77	:77	:78	:81	:81	:78
Commercially available in UK	: N	N	N	N	N	N	Y	N	N	N	N	N	N	N	Y	Y	N	N	N
Of specialised interest only	: Y						Y	Y											
Size of historic database (H=>100)	:13	?:	?:	H:	?:	?:	5:	?:	20	?:	?:	H:	H:	?:	H:	H:	?:	?:	?
Scope of historic database (C=company,G=several)	: C	C	C	?	C	C	C	?	C	?	C	?	G	?	G	G	C	C	G
Type of model (E=general eqn) (C=cost/instruction,H=heuristic,P=phenomenological)	: C	C	C	E	C	?	C	E	?	E	H	C	C	E	H	P	H	C	C
******MODEL NUMBER**********	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
SIZE																			
Source code	:		I			I	I					I	I			I	I	I	I
Object code	: I	I		I	I		I	I	I	I	I		I	I	I			I	I
Functions	:											I		I	I		I		
Data items	:		I			I			I	I		I					I		
Hardware configuration (peripherals)	:			I				I				I	I		I		I		I
No. of people needed	:					I					I								
Quantity of documentation	:		I	I		I		I					I						I
******MODEL NUMBER**********	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
PROGRAM ATTRIBUTES																			
Language	:		I						I		I	I		I		I			
Difficulty/application mix	: I		I	I	I	I			I		I	I	I	I	I	I	I		I
Complexity (interfaces)	:		I	I	I	I			I			I	I			I	I		I
Percent new	:				I			I	I		I	I		I	I	I	I		

OMPARISON OF COST MODELS

******MODEL NUMBER**********	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
PROJECT ATTRIBUTES																			
Volatility of requirement			I			I						I	I	I	I		I		I
Reliability/documentation				I	I	I			I			I	I		I				I
Security level						I			I			I					I		
******MODEL NUMBER**********	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
HARDWARE ATTRIBUTES																			
Time utilisation		I	I			I			I		I	I	I	I	I	I	I		I
Store utilisation		I	I			I			I			I	I	I	I	I	I		I
Concurrent HWare development					I	I			I			I	I				I		I
Host machine not target			I										I				I		I
Type of target computer							I						I	I	I			I	I
******MODEL NUMBER**********	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
DEVELOPER ATTRIBUTES																			
Quality of personnel			I		I	I			I			I					I		
Continuity of personnel			I	I		I											I		
Experience with hardware				I		I		I	I			I	I				I		I
Experience with application			I			I		I	I		I	I	I		I		I		I
Experience with language					I	I		I				I		I	I		I		
Software tools/techniques					I	I					I	I	I	I	I	I	I		I
Adequacy of devel. hardware						I					I		I						I
No. of development sites				I	I			I				I	I						I
Management style			I										I		I	I		I	I
******MODEL NUMBER**********	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
FINANCE AND SCHEDULE																			
Development schedule			I									I			I	I		I	
Resource Constraints						I			I			I	I	I	I	I			
Inflation rates												I			I				I
Manpower rates													I		I				I
Computer costs																			
Costs for support/overheads													I		I				I
******MODEL NUMBER**********	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
OUTPUTS PROVIDED BY THE MODEL																			
Secondary costs (computers)													O	O					O
Cost to develop	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
Cost of maintenance														O	O	O		O	
Cost(man hours=E, money=M	E	M	M	E	M	M	E	?	?	?	?	EM	EM	EM	EM	EM	EM	E	EM
Sensitivity analysis															O	O			
Development schedule															O	O		O	
Types of labour												O	O	O	O		O		O
Confidence levels												O				O	O		
******MODEL NUMBER**********	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

KEY

I= input to model

O= possible outputs from the model

APPENDIX C

HISTORIC DATABASE - DATA REQUIRED.

C.1 INTRODUCTION.

There are currently two software cost estimating models available in the U.K. It is desirable that any costing data which is collected be useable for checking these two models, and for comparing their performance, as well as being available for wider use. The data required by the two models is therefore listed below. It is likely that any cost estimating models developed in the future will require similar data.

C.2 DATA REQUIRED.

C.2.1 Project Size.

Size is required (for Putnam) in lines of deliverable source code. Size is input for each of the constituent parts of the software, with a best estimate and range for each module.

PRICE requires size in machine code instructions, or as one of the following sets of items from which size may be derived:

1. size, in source code instructions, plus an expansion factor for conversion to machine code instructions:
2. total number of functional modules (assuming 90 instructions/module):
3. level, the average level of the work breakdown structure, plus structure, an empirical variable derived from similar projects, relating level to the number of functional modules. This item will not normally be available on a historic project, but can be derived by PRICE-S from one project, for use with other similar projects.

C.2.2 Volatility Of The Requirement.

Expected percentage change in the requirement before delivery.

C.2.3 Application Mix.

A single parameter which summarizes the application mix of instructions. A table of typical values is given in the PRICE-S manual, varying from about 0.86 for a simple mathematical application to 10.95 for an operating system. Alternatively the actual instruction mix can be given, either as the actual number of instructions of each type, or as the fraction of instructions of each type. The instruction types are:

1. data storage and retrieval (weight 4.10):
2. on-line communications (weight 6.16):
3. time critical code (weight 10.95);
4. real time command and control (weight 8.46):
5. interactive operations (weight 10.95);
6. mathematical applications (weight 0.86);
7. string manipulation (weight 2.21);
8. operating systems (weight 10.95);
9. one user defined category (weight user defined).

If the application mix is given as a single parameter, the description from the above list which most nearly characterises the application should also be given, together with the percentage of time critical code.

C.2.4 Language.

1. percentage coded in a high level language:
2. principal language to be used;
3. use of database management system;
4. percentage of source statements written in database management language;
5. use of report writer;
6. percentage of source statements written in using the report writer.

C.2.5 Percentages Of New Design And Of New Code.

If the application mix is given as a single parameter the percentages of new design and of new code are also given as single parameters. If the application mix is divided into types, the the percentages of new design and of new code may also be given by application type;

C.2.6 Hardware/software Design Interactions.

The project must be described as one of the following:

1. new system with many interfaces. Must interact with external systems;

2. new system, but not interfacing much with other systems;

3. rebuild of existing system, large sections of existing logic will be used but the project will need new code and integration. Only minor enhancements are required;

4. composite project made up of a set of independent subsystems, but with a few interactions;

5. composite project made up of a set of independent subsystems with a minimum of interactions;

6. conversion of an existing system for a new target machine. Less than 15% new code:

C.2.7 Hardware Utilisation.

The percentage of the available store, and of the target processor time consumed, must be given.

C.2.8 Platform Type.

Platform type indicates the type of reliability, quality assurance, maintainabilty and documentation required. The platform values are selected by comparison with a table of typical values, published in the PRICE-S manual, ranging from 0.6 for internally developed, non deliverable software to 2.5 for a manned space craft.

C.2.9 Experience Of The Development Organisation

(minimal, average or extensive):

1. project personnel, quality and experience.
2. experience with the application;
3. experience with the language;
4. experience with the development computer;
5. hardware familiarity.

C.2.10 Characteristics Of The Development Organisation:

1. adequacy of the development tools (minimal, average or above average);
2. whether the project is multi national:
3. number of development locations.

C.2.11 Software Tools And Techniques Employed:

1. percentage of the development to be done or line:
2. percentage of the development computer dedicated to this project:
3. extent to which structured programming will be used (<25%,25%-75% or>75%);
4. extent to which design and code inspection will be used (<25%,25%-75% or>75%);
5. extent to which top- down development will be applied (<25%,25%-75% or>75%):
6. extent to which chief programmer teams will be used (<25%,25%-75% or>75%);
7. extent to which prototypes will be used (<25%,25%-75% or>75%):
8. whether MASCOT is used;
9. extent of the use of project database to assist in project control (<25%,25%-75% or>75%);

10. extent of the use of project libraries (<25%,25%-75% or>75%);

11. which formal design methods are in use;

C.2.12 The Project Development Schedule.

The project development schedule given as start and end dates of each of the three phases, design, implementation and test and integration. (For new projects only the start date is mandatory, as the model derives its own development schedule).

C.2.13 Costing Information.

The Putnam model requires the average cost per labour unit (manyear or manmonth) with standard deviation. This should be what is known as the fully burdened labour rate (i.e. the rate including overheads; costs of computer time and any other costs which the user wishes to have included in the cost estimates.

The PRICE model gets the same information from the resource parameter, indicating the performance characteristics of the development organisation. The factors which decide the value of the resource parameter are

1. management style of the development organisation;

2. computer operating charges;

3. labour and overhead rates.

which affect the cost rather than the development schedule of the project. The resource parameter varies from about 2.0 to 4.0 depending on the nature of the organisation. For a given organisation, the parameter can be derived using the PRICE-S model with historic data, in calibration mode.

C.2.14 Resource Constraints.

When a project is subject to constraints either of funding or of manpower, the constraints will affect the development schedule. The constraints which are given for each of the three project phases, are given either in terms of maximum number of labour units available per month (manhours or manmonths), or as a number pair giving maximum number of labour units per month together with the average cost of each labour unit.

C.2.15 Project Cost.

The cost should be given both in money terms (pounds) and in labour units (manhours or manmonths).

C.2.16 Start Year.

C.2.17 Other Data.

The following optional information may be used to cross check the data:

1. number of different types of data storage and retrieval devices and number of physical devices;

2. number of different types of on line communication devices and number of physical devices;

3. number of different types of real time command and control devices and number of physical devices;

4. number of different types of interactive communication devices and number of physical devices.

APPENDIX D

HISTORIC DATABASES IN THE USA

The following databases, collected in the USA, have been mentioned in the literature.

Organisation: Systems Development Corporation (SDC)
Year: 1964
No. of observations: 27
No. of variables: 98
Comments: SDC Phase 1, all SDC developments.

Organisation: Systems Development Corporation (SDC)
Year: 1965
No. of observations: 74
No. of variables: 63
Comments: SDC Phase 2, all SDC developments

Organisation: Systems Development Corporation (SDC)
Year: 1966
No. of observations: 169
No. of variables: 94
Comments: SDC Phase 3, 69 observations from Phase 2 plus 100 from other organisations.

Organisation: Systems Development Corporation (SDC)
Year: 1967
No. of observations: 22
No. of variables: 146
Comments: SDC Phase 4, all space software systems.

Organisation: Planning Research Corporation (PRC)
Year: 1970
No. of observations: 18
No. of variables: 84
Comments: PRC 1, 16 ADP developments, plus one embedded space system

Organisation: Planning Research Corporation (PRC)
Year: 1970
No. of observations: 20
No. of variables: 93
Comments: Also called ADREP Database, US Army ADP systems

Organisation: IBM
Year: 1975
No. of observations: 11
No. of variables: 8
Comments: IBM developments, military systems

Organisation: IBM
Year: 1975
No. of observations: 24
No. of variables: 2
Comments: Unpublished, the only variables were man months and size in source instructions. Embedded weapon systems.

Organisation: Logicon Inc.
Year: 1968
No. of observations: 7
No. of variables: 37
Comments: Benchmark type problems, single programmer per project.

Organisation: John Hopkins University
Year: 1975
No. of observations: 39
No. of variables: 4
Comments: Study to define software problems facing DoD.

Organisation: Rome Air Development Center (RADC)
Year: ?
No. of observations: approx 400
No. of variables: ?
Comments: Collected from a wide variety of sources, mainly ground based management information systems.

Organisation: USAF Data Systems Design Center (DSDC) -PARMIS data
Year: 1978
No. of observations: 17
No. of variables: ?
Comments: collected and analysed by GRC.
Organisation: Goddard Space Flight Center
Year: ?
No. of observations: 7
No. of variables: ?
Comments: mostly Fortran, real-time programs.

APPENDIX E

REFERENCES

1. M.Kayton "Keeping engineering within budget" Technology Review M.I.T. Cambridge MA Jan 76

2. G.Sandler and B.Rachowitz "Software Cost Models - Grumman Experience" Quantitative Software Model Conference IEEE 1979.

3. H.Dircks ""SOFCOST" Grumman's software cost estimating model" 1981 IEEE NAECON

4. C.E.Walston and C.P.Felix IBM "A Method of Programming Estimation" IBM Systems Journal Vol 16 no. 1 1977

5. J.D.Aron IBM "Estimating resources for large programming systems" NATO Conference on Software Engineering Techniques, New York 1969.

6. E.N.Dodson et al. GRC "Advanced Cost Estimating and Synthesis Techniques for Avionics" Final Report CR-2-461, 1975

7. R. Thibodeau and E.N.Dodson GRC "The implications of life cycle phase interrelationships for software cost estimating" 1978 Software Life cycle Management conference.

8. C.A.Graver and W.M.Carriere GRC "Cost reporting elements and activity cost tradeoffs for defense" May 77 ESD-TR-77-262-VOL-1

9. R.Thibodeau GRC "An evaluation of software cost estimating models" June 81 RADC-TR-81-144

10. M.Finfer and R.Mish "Software acquisition management guidebook" ESD-TR-78-140

11. G.A.Bourdon and J.A.Duquette ESD "A computerized model for estimating software life cycle costs" Apr 78 ESD-TR-235-Vol 1

12. B.C.Frederick Tecolote Research Inc., "A Provisional model for Estimating computer program development costs", TM-7/Rev 1, Dec 1974

13. R.W.Wolverton TRW "The cost of developing large scale software" IEEE Trans. on Computer, Vol C-23, no. 6

14. B.W.Boehm and R.W.Wolverton TRW "Software cost modelling- some lessons learned" Journal of Systems and Software 1, 1980

15. D.Ferens and R.Harris, Air Force Avionics Laboratory "Avionics Computer Software operation and support cost estimation" NAECON 1979 Proceedings

16. D.Ferens and T.James, Air Force Avionics Laboratory "Application of the RCA PRICES software cost estimation model to Air Force Avionics Laboratory Programs" Final Technical report AFAL-TR-79-1164 Wright Patterson AFB

17. T.James, Air Force Avionics Laboratory "Software Cost Estimating methodology" Final technical report AFAL-TR-77-66

18. F.Freiman and R.Park, RCA "The PRICE software cost model" NAECON May 1979

19. C.Mauro , RCA,"PRICE S RESO and CPLX calibration" RCA PRICE Systems

20. C.Mauro , RCA,"PRICESL" RCA PRICE Systems 1981 IEEE

21. RCA PRICE-S and PRICE-SL Reference manuals

22. R.E.Steffey "An analysis of the RCA PRICES software cost estimation model as it relates to current air force computer software acquisition and management" Dec 1979 AFIT/GSM/SM/79D-20

23. M.Zelkowitz , University of Maryland,"Resource Estimation for medium scale software projects" Proceedings of the interface symposium on computer Science and statistics Ontario May 1979

24. W.H.Walker "an approach to software life cycle cost modelling" AFIT/GCS/EE/78-21 Dec 78

25. J.H.Herd et al. Doty Associates Inc. "Software Cost Estimation Study Study results Vol 1" RADC-TP-77-220-Vol-1.

26. P.J.Nelson et al. Doty Associates Inc. "Software Cost Estimation Study Study results Vol 2" RADC-TR-77-220-Vol 2.

27. J.A.Clapp, Mitre Corp. "A Review of software cost estimation methods" ESD-TR-76-271

28. R.C.Tausworthe "Deep Space Network software cost estimation model" Apr 1981 NASA-CR-164277

29. R.C Tausworthe "The work breakdown structure in software project management" Journal of systems and software 1, 1980.

30. L.Putnam and A Fitzsimmons "Estimating Software Costs Parts 1,2 and 3" Datamation Sept to Dec 1979

31. L. Putnam "SLIM"

32. L.Putnam "Example of an early sizing, cost and schedule Estimate for an application software system" Proc. COMPSAC 78.

33. L.Putnam "A general empirical solution to the macro software sizing and estimating problem" IEEE Trans. on Software Engineering July 78.

34. L.Putnam "The real economics of software development" Symposium on the Economics of information processing, Dec 1980.

35. L.Putnam "Software costing and life cycle control" Workshop on quantitative models for reliability, complexity and cost, Oct 1979, New York.

36. L.Putnam "Measurement data to support sizing estimating and control of the software life cycle" COMPCON Spring 78

37. L.Putnam "Progress in modelling the software life cycle in a phenomenological way to obtain quality estimates and dynamic control of the process" Proc. second software life cycle management workshop 1978.

38. W.Myers "A statisitical approach to scheduling software development" Computer Dec 78

39. J.Durway "Sensitivity Analysis and simulation" Infosystems May 79

40. P.Norden "Useful tools for project management" Operations Reseach and development 1963

41. F.Brooks "The Mythical man month" Addison Wesley Publishing Co. 1975.

42. R.Jensen , Hughes Aircraft Co.,"A macro level software development cost estimation methodology" 1980 Ciruits systems and computers, ASILOMAR Conference

43. E.Nelson , TRW,"Developing a software cost methodology" TRW IEEE Computer Society International conference, Washington Sept 76

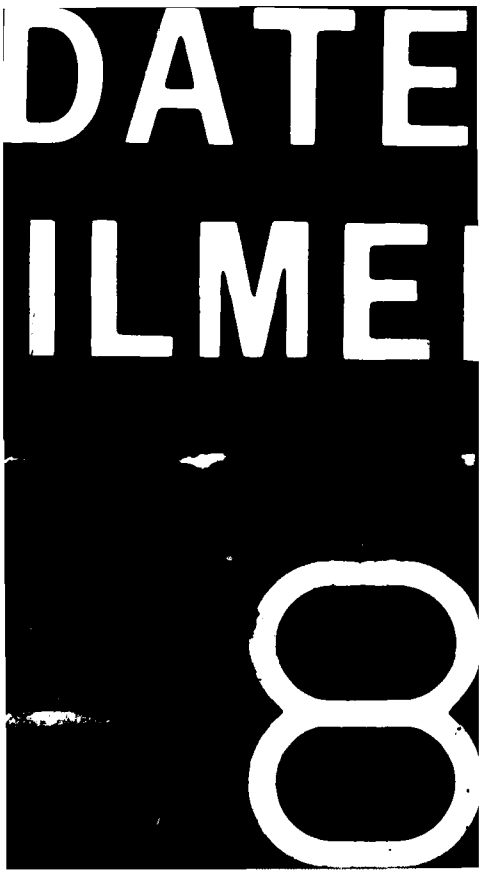
44. J.Bailey and V.Basili , University of Maryland,"A Meta-model for software development resource expenditures" 1981 Software Engineering Conf. Proceedings

45. A.Ferrentino "Making software development estimates "good"", Datamation Sept 81.

46. J.Golden et al. "Software cost estimating: craft or witchcraft", Xerox Corporation, Data Base Spring 1981.

47. M.H.Halstead "Elements of software science", Elsevier, North Holland.

48. A.L.Kustanowitz, IBM "System life cycle estimation (SLICE): A new approach to estimating resources for application program development.", IEEE First International Computer Software and Applications Conference, Chicago.


DATE
ILME
8